AF

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant : Todor Georgiev
Serial No. : 09/996,200
Filed : November 28, 2001
Title : A TOOL FOR EXTRACTING AND MANIPULATING COMPONENTS OF WARPING TRANSFORMS

Art Unit : 2672
Examiner : Ryan R. Yang

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

TRANSMITTAL OF SUPPLEMENTAL APPEAL BRIEF

Responsive to the Notification of Non-Compliant Appeal Brief mailed February 10, 2006, the applicant encloses a Supplemental Appeal Brief to provide the materials and information found to be absent. In particular, the Supplemental Appeal Brief includes: (9) Evidence Appendix; and (10) Related Proceedings Appendix.[1]

The applicant submits that the Supplemental Appeal Brief is now fully compliant.

Respectfully submitted,

Date: 3/8/06

Mandy Jubang
Reg. No. 45,884

Customer No. 021876
Telephone: (617) 542-5070
Facsimile: (617) 542-8906

[1] The applicant encloses a copy of the postcard receipt which itemizes and identifies the items that were submitted with the Appeal Brief on October 7, 2004, one of which was the Evidence Appendix, including a marked up version of Thomas reference (10 pages). As noted, a replacement copy of the Evidence Appendix is enclosed.

CERTIFICATE OF MAILING BY FIRST CLASS MAIL

I hereby certify under 37 CFR §1.8(a) that this correspondence is being deposited with the United States Postal Service as first class mail with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

March 8, 2006
Date of Deposit

Cassie Chandler
Signature

Cassie Chandler
Typed or Printed Name of Person Signing Certificate

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicant : Todor Georgiev
Serial No. : 09/996,200
Filed : November 28, 2001
Title : A TOOL FOR EXTRACTING AND MANIPULATING COMPONENTS OF WARPING TRANSFORMS

Art Unit : 2672
Examiner : Ryan R. Yang

Mail Stop Appeal Brief - Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

SUPPLEMENTAL APPEAL BRIEF

(1) Real Party in Interest

Adobe Systems Incorporated

(2) Related Appeals and Interferences

None known.

(3) Status of Claims

Claims 1-39 are pending in the case. (See Claims Appendix). Claims 1-3, 10-11, 13-18, 25-26, 28-32, 34-35, and 37-39 were rejected under 35 U.S.C. 102(b) as having been anticipated by Thomas et al. (ACM, Nov 1995). Claims 4, 19 and 36 were rejected under 35 U.S.C. § 103(a) as having been unpatentable over Thomas et al. in view of Reyzin (U.S. 6,215,915). Claims 5 and 6 were rejected under 35 U.S.C. § 103(a) as having been unpatentable over Thomas in view of Foley et al (Computer Graphics: Principles and Practice, 2nd Edition). Claims 20-24 were rejected under 35 U.S.C. § 103(a) as having been unpatentable over Thomas and Reyzin in view of Foley. Claims 12, 27, and 33 were rejected under 35 U.S.C. § 103(a) as

CERTIFICATE OF MAILING BY FIRST CLASS MAIL

I hereby certify under 37 CFR §1.8(a) that this correspondence is being deposited with the United States Postal Service as first class mail with sufficient postage on the date indicated below and is addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450.

March 8, 2006
Date of Deposit

Cassie Chandler
Signature

Cassie Chandler
Typed or Printed Name of Person Signing Certificate

having been unpatentable over Thomas in view of Choi et al. (U.S. 6,157,750). Claims 7-9 were objected to as being dependent upon a rejected base claim, but would have been allowable if rewritten in independent form including all of the limitations of the base claim and any intervening claims. All of the pending claims are being appealed.

(4) Status of Amendments

No substantive amendments have been made since the final office action dated March 13, 2004.

(5) Summary of Claimed Subject Matter

A variety of computer programs, such as Adobe® Photoshop® available from Adobe Systems Incorporated, provide tools that aid a user in distorting an image for visual effect. [specification, page 1, lines 7-9] Initially, an unaltered image (also referred to as a "source image") is associated with a distortion grid having zero distortion vectors. [specification, page 1, lines 13-14] Once the user applies a distortion to the source image, the altered image is referred to as a distorted image, which is associated with a distortion grid having a distortion vector. [specification, page 1, lines 11-12] Collectively, the distortion vectors indicate how the distorted image can be manipulated to obtain the source image. [specification, page 1, lines 12-13] As each distortion is sequentially applied to the distorted image, distortion vectors are added to the distortion grid and summed with the distortion vectors already in the distortion grid. [specification, page 1, line 16] In other words, the distortion vectors in the distortion grid at any point in time provide a sum of all previous distortion vectors, without the history of the individual distortions themselves. [specification, page 1, lines 16-18] Each distortion vector has one or more of the following components: translation, magnification, rotation, directional scaling, and skew. [specification, page 2, line 30 – page 3, line 1]

The appellant's claimed subject matter relates to techniques for extracting one or more components of a distortion local to a first area of a distorted image, and applying the extracted components to a second area of the distorted image. [specification, page 3, lines 4–5] A user uses a tool to select: (1) an area of a distorted image; and (2) one or more components to be extracted from the selected area. [specification, page 4, lines 19-26] The tool uses a plurality of

points local to the selected area to approximate an affine transformation that represents the distortion vectors local to the selected area. [specification, page 5, line 1 – page 6, line 14] The tool then manipulates the affine transformation to extract a matrix representing the user-selected components. [specification, page 6, line 19 – page 11, line 2] In response to a user action, the extracted matrix is applied by the tool to a second area of the distorted image to further distort the image. [specification, page 11, line 6 – page 12, line 3]

Claim 1 recites a method including: in response to user action on a canvas, selecting at least one area of a first image which relates to an area on a distortion grid [specification, page 4, lines 23-26]; using a plurality of points local to the at least one area to calculate a distortion [specification, page 4, line 27 – page 6, line 18]; extracting at least one component of the distortion [specification, page 6, line 19 – page 11, line 2]; and applying the at least one component to a second area of the first image [specification, page 11, line 4 – page 12, line 3].

Claim 16 recites a computer program product, disposed in a computer readable medium, having instructions to cause a computer to use a plurality of points surrounding a first area of an image related to an area in a distortion grid to calculate at least one component of a distortion at the first area [specification, page 4, line 23 – page 11, line 2]; and apply the at least one component of the distortion to a second area of the image [specification, page 11, line 4 – page 12, line 3].

Claim 31 recites a computer program product having instructions stored in a computer readable medium, containing instructions to cause a computer to display a first image on a canvas, the first image being related to an area on a distortion grid [specification, page 4, lines 13-22]; responsive to an input device controlled by a user, select an area of the first image [specification, page 4, lines 23-26]; responsive to a selection by the user from a menu, extract at least one component of a distortion from the area [specification, page 6, line 19 – page 11, line 2]; and responsive to movement and location of a cursor controlled by the user, apply the at least one component to a second area of the first image [specification, page 11, line 4 – page 12, line 3].

(6) Grounds of Rejection to be Reviewed on Appeal

The grounds of rejection to be reviewed on appeal are:

1. Did the examiner properly reject claims 1-3, 10-11, 13-18, 25-26, 28-32, 34-35, and 37-39 under 35 U.S.C. 102(b) as being anticipated by Thomas et al. (ACM, Nov 1995)?
2. Did the examiner properly reject claims 4, 19, and 36 under 35 U.S.C. § 103(a) as being unpatentable over Thomas et al. in view of Reyzin (U.S. 6,215,915)?
3. Did the examiner properly reject claims 5 and 6 under 35 U.S.C. § 103(a) as being unpatentable over Thomas in view of Foley et al (Computer Graphics: Principles and Practice, 2nd Edition)?
4. Did the examiner properly reject claims 20-24 under 35 U.S.C. § 103(a) as being unpatentable over Thomas and Reyzin in view of Foley?
5. Did the examiner properly reject claims 12, 27, and 33 under 35 U.S.C. § 103(a) as being unpatentable over Thomas in view of Choi et al. (U.S. 6,157,750)?

(7) Argument

(a) Anticipation

For a reference to anticipate a claim, each element and limitation of the claim must be found in the reference. *Hoover Group, Inc. v. Custom Metalcraft, Inc.*, 66 F.3d 299, 302 (Fed. Cir. 1995). Thomas does not disclose all of the elements of the claims. For reference, a marked up version of Thomas including reference letters is provided in the appendix.

The Thomas approach

For ease of reference, a marked up version of Thomas is provided in the Evidence Appendix.

Thomas describes how effects based on cartoon animation principles, such as the principles of solidity, exaggeration, reinforcement, attachment, and reluctance, can be used to enhance the illusion of direct manipulation by strengthening the impression that users are manipulating "real" objects presented in a graphical interface. [Thomas, page 4 at "A" and page

11 at "B"]. Thomas discloses techniques that application programmers can use to implement such animation effects in interfaces. [Thomas, page 3 at "C"]

In examples described in Thomas, the techniques are implemented in a drawing editor that supports the creation of simple figures, such as lines and polygons, and allows simple editing operations, such as moving, scaling, and rotating, to be performed on the figures. [Thomas, page 4 at "D"] Figures 2, 3, 4, 5 and 6 (reproduced below for reference) provide examples of user interactions with the drawing editor before and after the techniques are implemented.

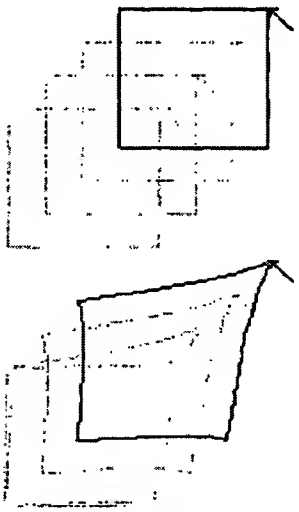


Figure 2: Animating a move operation

Figure 2 shows how the editor animates a move operation on a rectangle. Of course, these static images cannot fully convey the dynamic feel of the interaction: we have superimposed several successive frames from the interaction to suggest the effect. The top of the figure shows the operation without the animation effect; the rectangle merely moves to follow the mouse. The bottom shows the operation with animation; note how the corner of the rectangle stays attached to the mouse point while the bulk of the object lags slightly behind. In operation, the effect is that of manipulating a heavy "rubbery" object that distorts as it is pushed and pulled.

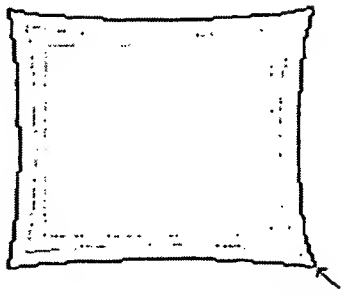


Figure 3: Animating a scaling operation

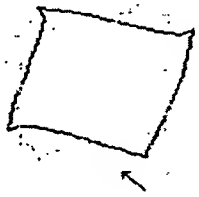


Figure 4: Animating a rotate operation

The same kind of effect works for other common editor operations. For example, figure 3 illustrates a scaling operation, and figure 4 shows a rotation. In each case, the editor uses the same strategy to animate the interaction; the part of the object that is "grabbed" is controlled by the mouse, while the bulk of the object lags behind. (For the scaling operation, we use an effect that suggests that all corners are simultaneously under the control of the mouse.)

The editor also supports simple constraining effects that show how animation can convey extra information about the interaction. Consider, for example, an attempt to move an object that is fixed in place. One response to this attempt might simply be to prevent the object from following the mouse. However, this lack of visible feedback might be misinterpreted as the result of a failure to "grasp" the object correctly. A user might make several attempts at the operation before realising the true cause of the lack of response. Another strategy might be to allow the object to follow the mouse, but then to snap it back to its original place when released. This approach avoids the problem with lack of feedback, but can lead to surprises when a carefully placed object suddenly jumps back.

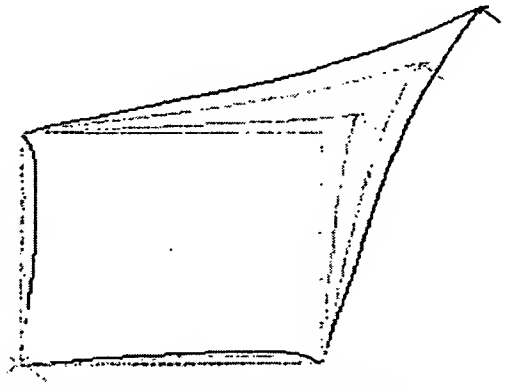


Figure 5: Attempting to move a pinned object

Figure 5 shows how our editor avoids both problems using animation. As the user attempts to drag the pinned object, the grab point stays attached to the mouse but the bulk of the object stays fixed. The effect is as if user is pulling on a corner of an object that is anchored in place. The feedback makes it clear that the user is attempting to move the object, but that the attempt will not succeed. When the grab point is released, the object will spring back to its original shape.

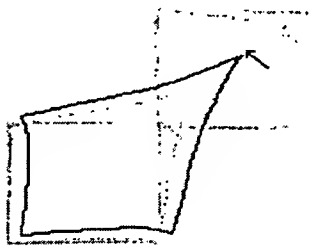


Figure 6: Moving under the influence of "gravity"

The same effect can suggest other constraints on objects. For example, figure 6 shows the effect of moving an object under the influence of a "gravity" field that causes the object to snap to a regular grid. As the user pulls the object away from a grid point, the grasped corner stretches while the object stays fixed. When pulled beyond a certain distance, the gravity suddenly "lets go" and the object snaps to the next grid point. Again, the feedback makes it clear what the current state of the interaction is, what the user must do to achieve the desired result, and what will happen if the object is released at any time.

To create the effect of animated interactions, Thomas describes creating and storing a number of warping transformations in the drawing editor. [Thomas, page 6 at "E"] Each warping transformation is associated with a particular operation and can be characterized by a set of bound vectors (also referred to as "warp vectors") that describe the transformations applied to key points in the coordinate space. [Thomas, page 6 at "F"] Transformations for points that do not coincide with vectors are calculated by interpolating between the vectors. [Thomas, page 6 at "F"]

To suggest the effect of moving a rectangle, for example, the drawing editor selects the warping transformation associated with the move operation from the set of stored warping transformations, and computes the warp vectors for the move operation (using the formula reproduced below) for each frame of the interaction. The drawing editor applies the warp vectors to the coordinate system of the drawing space (as shown in Figures 11 and 12 reproduced below), and then draws the object as a simple rectangle on the warped coordinate system of the drawing space. When the rectangle is displayed on the screen, it appears distorted (as shown in Figure 8 reproduced below).

For example, consider the warp vectors for a move operation (figure 12). Since $|p_1| = 0$ and $p_2 = p_3 = p_4 = p$, the computation of the transformation simplifies as follows:

$$\begin{aligned} q' &= q + \sum_{i=1}^{n} w_{qi} p_i \\ &= q + w_{q1} p_1 + \sum_{i=2}^{n} w_{qi} p_i \\ &= q + 0 + p \sum_{i=2}^{n} w_{qi} \\ &= q + p(w_{q2} + w_{q3} + w_{w4}) \end{aligned}$$

```
Transformer t;
Warp w;
w.add_vector(100, 80, 0, 0);
w.add_vector(0, 80, -10, -10);
w.add_vector(0, 0, -10, -10);
w.add_vector(100, 0, -10, -10);
t.warp(&w);
```

Figure 11: Applying a warp for a move operation

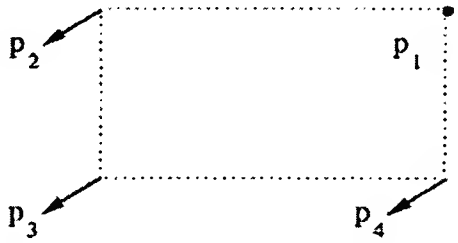


Figure 12: Symmetry of Translation Vectors

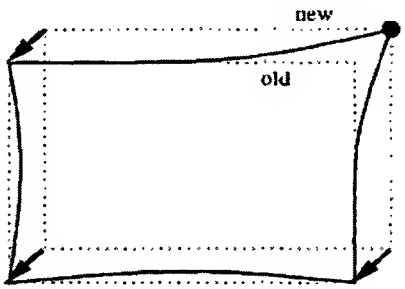


Figure 8: Calculating warp vectors for move

Group I (claims 1, 2, 10, 11, 16, 17, 25, 26, 31, 32, and 34)

For the purposes of this appeal only, claims 1, 2, 10, 11, 16, 17, 25, 26, 31, 32, and 34 rise and fall together. Claim 1, which is representative of this group, recites:

> **A method comprising:**
> **in response to user action on a canvas, selecting at least one area of a first image which relates to an area on a distortion grid;**
> **using a plurality of points local to the at least one area to calculate a distortion;**
> **extracting at least one component of the distortion; and**
> **applying the at least one component to a second area of the first image.**

In all of the examples described in Thomas, there is no mention of using a plurality of points local to an area of an image to calculate a distortion, extracting a component of the distortion, and applying the extracted component to a different area of the image.

As previously discussed, Thomas is directed to techniques for animating a user interaction with objects displayed in a graphical interface of an application. The Thomas approach involves storing warping transformations in a drawing editor and applying an appropriate warping transformation to an object displayed on a graphical interface in order to provide visual animated feedback to a user. There is no mention of calculating a distortion local to an area of an image, extracting a component of the distortion, and applying the extracted component to another area of the image.

The examiner's rejection of claim 1 as set forth in the Office Action mailed on March 12, 2004 is reproduced below for reference:

> **As per claim 1 ,Thomas et al., hereinafter Thomas, discloses a method comprising:**
> **in response to user action on a canvas, selecting at least one area of a first image which relates to an area on a distortion grid (Figure 3 where a corner of the object is grabbed for scaling);**
> **using a plurality of points local to the at least one area to calculate a distortion local to the area (Figure 3 where the area close to the selected corner is distorted; the distortion is not limited to one point but to a plurality of points local to the area);**
> **extracting at least one component of the distortion (the scaling factor for Figure 3); and**
> **applying the at least one component to a second area of the first image (Figure 3 where the other corners are also distorted).**

The appellant interprets the examiner's remarks with respect to claim 1 as saying that the drawing editor calculates an amount by which the corner of the object that is grabbed for scaling

is distorted, and applies this scaling amount to the other corners of the object. However, this is not what is described in Thomas.

Figure 3 and the related text say nothing about using a plurality of points local to an area of an image to calculate a distortion. Figure 3 of Thomas and the related text describe animating a scaling operation in which all four corners of a rectangle are simultaneously under the control of a mouse when one corner is dragged by a user of the mouse. Although these portions of Thomas neither disclose nor suggest the detail of how the animation of the scaling operation is performed, the appellant presumes that the techniques used are similar to those previously discussed in relation to animating a move operation. In other words, the drawing editor selects the warping transformation associated with the scaling operation from the set of stored warping transformations, computes the warp vectors for the scaling operation for each frame of the interaction, applies the computed warp vectors to the coordinate system of the drawing space, and draws the object on the warped coordinate system of the drawing space. When the object is displayed on the screen, it appears to be scaling outwards in the case of Figure 3.

In the Advisory Action mailed on May 3, 2004, the examiner responded to the appellant's remarks provided in the Reply to Action of March 12, 2004 as follows:

> **examiner notes in applicant's specification, applicant explains "distortion may be thought of as a grid of vectors ("the distortion grid"). Therefore, the grid is really a plurality of vectors. This is similar to Thomas' warp vectors (page 6, column 2, paragraph 4). Thomas also discloses distortion has components (Figure 9 has subroutine for rotation, scaling, and translation).**

The appellant disagrees with the examiner's assertion that the appellant's "distortion grid" is similar to Thomas' "warp vectors". As stated in appellant's specification at page 1, lines 11-13:

> **A distortion may be thought of as a grid of vectors (the "distortion grid") with each vector corresponding to a single point in an image. The vectors indicate how the image is modified to obtain the original, undistorted source image.**

To the contrary, Thomas describes the warp vectors as being used to add animation to a user interaction with an object. The warp vectors are calculated and applied to an underlying coordinate system in order to effect a particular warp transformation. Thomas says nothing about the warp vectors being a distortion grid that indicates how an image can be modified to

obtain an original, undistorted image. Rather, Thomas describes applying the warp vectors in order to distort an image.

Arguably, the matrix formed by the application of successive affine transformations on an object in Thomas' system might be interprested as the appellant's "distortion grid". On page 8, Thomas states:

> Conceptually, a Transformer represents a succession of discrete mapping steps applied one after the other. For example, a Transformer might represent a scaling followed by a translation followed by a rotation. One characteristic of affine mappings is that they can be computed as matrix products in a homogeneous coordinate system; as a consequence, any sequence of affine mappings can be represented as a single matrix operation. Because of this property, the original Transformer stored only a single 3 × 2 matrix; successive transformations on the Transformer modified the matrix to reflect the new aggregate operation.
>
> However, the warp mapping is not affine, and so combinations of transformations that involve warping cannot be represented as a single matrix. We modified the implementation of the Transformer class so that it keeps a list of transformation items; successive affine transformations are combined into a single matrix item as before, while warp transformations are represented by individual warp items.

Even though Thomas describes the types of affine transformations, such as rotation, scaling, and translation, that may be applied to an object through the interface provided by the drawing editor, nowhere does Thomas describe extracting a rotation component or a scaling component or a displacement component from a distortion once the transformation has been applied, as required in claim 2. The 3x2 affine transformation matrix in Thomas suffers from the same drawbacks as the distortion grids described in the background of the appellant's specification, in that what exists in the 3x2 matrix represents the aggregate operation that has been performed on the original object, and does not reveal the history of the individual discrete mapping steps themselves.

Thus, Thomas neither discloses nor suggests that there be a calculation of a distortion, or that points local to the dragged corner be used in such a calculation. Nor does Thomas disclose or suggest extracting a component of the distortion. And Thomas does not disclose or suggest that it is such a component that is to be applied to a second area of an image. Accordingly, claims 1, 2, 10, 11, 16, 17, 25, 26, 31, 32, and 34 are not anticipated by Thomas.

Group II (claims 3, 18, and 35)

For the purposes of this appeal only, claims 3, 18, and 35 rise and fall together. Claim 3, which is representative of this group, recites:

The method of claim 2 wherein the extracting comprises calculating an affine transform from the plurality of points.

The examiner's rejection of claim 3 as set forth in the Office Action is reproduced below for reference:

> **As per claim 3, Thomas demonstrated all the elements as applied to the rejection of dependent claim 2, supra, and further discloses the extracting comprises calculating an affine transform from the plurality of points (Figure 3 where scaling is one component of affine transformation and "The original Transformation object supported only affine transformation such as rotation, scaling, and translation", page 7, last paragraph).**

The cited portions of Thomas describe affine transformations that can be applied to an object. There is no mention of calculating an affine transform from a plurality of points in the cited portion of Thomas or in any other part of the Thomas reference. Thomas does not disclose at least this feature of claims 3, 18, and 35.

Group III (claims 13, 28, and 37)

For the purposes of this appeal only, claims 13, 28, and 37 rise and fall together. Claim 13, which is representative of this group, recites:

The method of claim 1 wherein the applying is to an entire image.

The examiner's rejection of claim 13 as set forth in the Office Action is reproduced below for reference:

> **As per claim 13, Thomas demonstrated all the elements as applied to the rejection of independent claim 1, supra, and further discloses the applying is to an entire image (Figure 1 where the entire area can be applied).**

It is unclear from the examiner's objections to claim 13 what is meant by "Figure 1 where the entire area can be applied." Figure 1 of Thomas shows a screen image of the drawing editor being used to move an objection. Neither the cited portion of Thomas, nor any other portion of Thomas describe extracting a component of a distortion, much less applying the extracted component of the distortion to an entire image. Thomas does not disclose at least this feature of claims 13, 28, and 37.

Group IV (claims 14, 15, 29, 30, 38 and 39)

For the purposes of this appeal only, claims 14, 15, 29, 30, 38, and 39 rise and fall together. Claim 14, which is representative of this group, recites:

> **The method of claim 1 wherein the applying is to a second image.**

The examiner's rejection of claim 14 as set forth in the Office Action is reproduced below for reference:

> **As per claim 14, Thomas demonstrated all the elements as applied to the rejection of independent claim 1, supra, and further discloses the applying is to a second image ("The editor supports the creation of simple figures such as lines and polygons", page 4, line 43-44).**

The cited portion of Thomas describes the manner in which the drawing editor can be used to create a figure and editing operations (e.g., moving, scaling, and rotating) that may be performed with respect to the created figures. Neither the cited portion of Thomas, nor any other portion of Thomas describe extracting a component of a distortion from an area of an image, much less applying the extracted component of the distortion to a second image. Thomas does not disclose at least this feature of claims 14, 15, 29, 30, 38, and 39.

(b) Obviousness

It is well established that there must be some logical reason apparent from the evidence of record to justify combination or modification of references. *In re Regal*, 526 F.2d 1399 188, U.S.P.Q.2d 136 (C.C.P.A. 1975). In addition, even if all of the elements of claims are disclosed in various prior art references, the claimed invention taken as a whole cannot be said to be obvious without some reason given in the prior art why one of ordinary skill in the art would have been prompted to combine the teachings of the references to arrive at the claimed invention. Id. Even if the cited references show the various elements suggested by the examiner in order to support a conclusion that it would have been obvious to combine the cited references, the references must either expressly or impliedly suggest the claimed combination or the examiner must present a convincing line of reasoning as to why one skilled in the art would have found the claimed invention obvious in light of the teachings of the references. *Ex Parte Clapp*, 227 U.S.P.Q.2d 972, 973 (Board. Pat. App. & Inf. 1985).

Group V (claims 4, 19 and 36)

Claims 4, 19 and 36 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Thomas in view of Reyzin. For the purposes of this appeal only, claims 4, 19 and 36 rise and fall together. Claim 4, which is representative of this group, recites:

> **The method of claim 3 wherein the extracting further comprises decomposing the affine transform into a translation and a linear transform matrix.**

The examiner's rejection of claim 4 as set forth in the Office Action is reproduced below for reference:

> **As per claim 4, Thomas demonstrated all the elements as applied to the rejection of dependent claim 3, supra.**
>
> **Thomas discloses a method of distorting an area of an image using affine transformation. It is noted that Thomas does not explicitly disclose the extracting further comprises decomposing the affine transform into a translation and a linear transform matrix, however, this is known in the art as taught by Reyzin. Reyzin discloses a method of transformation in which the affine transformation is decomposed into a translation part and a linear transform matrix (column 3, line 15-30 where (Xo,Yo) is the translation part and M is the transform matrix).**

Reyzin discloses techniques for concurrently rotating, scaling, translating, skewing, shearing, or otherwise transforming an image via a sequence of one-dimensional transformations. [Reyzin, Abstract] Reyzin describes providing methods for general affine transformation of an image in two dimensions by generating an intermediate image via affine transformation of the source along a first axis, and then subjecting the intermediate image to affine transformation along a second axis. [Reyzin, column 2, lines 37-49]

One technique involves: (1) determining a mapping between coordinates in an intermediate image and those in a source image; (2) determining an intensity value for the pixel at each coordinate in the intermediate image; and (3) determining a mapping between coordinates in a destination image and those of the intermediate image. [Reyzin, column 2, line 50 – column 3, line 12] The resultant affine transformation is provided in the cited portion of

Reyzin (reproduced below for reference).

> According to further aspects of the invention, general affine transformations are effected in accord with the mathematical relation:
>
> $$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = M \cdot \begin{bmatrix} x_s \\ y_s \end{bmatrix} + \begin{bmatrix} x_o \\ y_o \end{bmatrix}$$
>
> $$M = \begin{bmatrix} m_{11} & m_{12} \\ m_{21} & m_{22} \end{bmatrix}$$
>
> where
>
> (x_d, y_d) represents a coordinate in the destination image;
> (x_s, y_s) represents a coordinate in the source image;
> (x_o, y_o) is an offset to be effected by the transformation; and
> M is a transformation matrix.

The Reyzin and Thomas references do not individually or in combination disclose or suggest extracting a component of a distortion by calculating an affine transform from a plurality of points, much less decomposing the affine transform into a translation and a linear transform matrix. Accordingly, claims 4, 19, and 36 are patentable over Thomas in view of Reyzin.

Group VI (claims 5 and 20)

Claim 5 is rejected under 35 U.S.C. § 103(a) as being unpatentable over Thomas in view of Foley. Claim 20 is rejected under 35 U.S.C. § 103(a) as being unpatentable over Thomas and Reyzin in view of Foley. For the purposes of this appeal only, claims 5 and 20 rise and fall together. Claim 5, which is representative of this group, recites:

> **The method of claim 3 wherein the extraction of magnification comprises calculating the determinant of a linear transform matrix.**

The examiner's rejection of claim 5 as set forth in the Office Action is reproduced below for reference:

> **As per claim 5, Thomas demonstrated all the elements as applied to the rejection of dependent claim 3, supra.**
>
> **Thomas discloses a method of distorting an area of an image using affine transformation. It is noted that Thomas does not explicitly disclose the extraction of magnification comprises calculating the determinant of a linear transform matrix, however, this is known in the art as taught by Foley et al., hereinafter Foley. Foley discloses that "the determinant of the matrix tells us ... how much the cube is expanded or contracted by the transformation (page 1104, line 2-3).**

The cited portion of Foley describes calculating a determinant of a matrix to determine a volume change resulting from a transformation. However, the Thomas and Foley references do not individually or in combination disclose or suggest extracting a magnification component of a distortion by calculating an affine transform from a plurality of points and calculating the determinant of a linear transform matrix. Accordingly, claims 5 and 20 are patentable over the cited references.

Group VII (claims 6 and 21)

Claim 6 is rejected under 35 U.S.C. § 103(a) as being unpatentable over Thomas in view of Foley. Claim 21 is rejected under 35 U.S.C. § 103(a) as being unpatentable over Thomas and Reyzin in view of Foley. For the purposes of this appeal only, claims 6 and 21 rise and fall together. Claim 6, which is representative of this group, recites:

> **The method of claim 3 wherein the extraction of rotation comprises calculating an angle from the elements of a linear transform matrix.**

The examiner's rejection of claim 6 as set forth in the Office Action is reproduced below for reference:

> **As per claim 6, Thomas demonstrated all the elements as applied to the rejection of dependent claim 3, supra.**
>
> **Thomas discloses a method of distorting an area of an image using affine transformation. It is noted that Thomas does not explicitly disclose the extraction of rotation comprises calculating an angle from the elements of a linear transform matrix, however, this is known in the art as taught by Foley. Foley discloses that in affine transformation, an angle of rotation from the transformation can be derived (page 203, the whole page).**

The cited portion of Foley describes deriving a rotation equation. However, the Thomas and Foley references do not individually or in combination disclose or suggest extracting a rotation component of a distortion by calculating an angle from the elements of a linear transform. Accordingly, claims 6 and 21 are patentable over the cited references.

Group VIII (claims 22, 23, and 24)

Claims 22-24 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Thomas and Reyzin in view of Foley. For the purposes of this appeal only, claims 22-24 rise and fall together. Claim 22, which is representative of this group, recites:

> **The computer program product of claim 19 wherein the one component of the distortion is a scaling amount, and the instructions to cause the computer to decompose the affine transformation further comprise instructions to calculate a pair of eigenvalues of the linear transform matrix, and wherein each eigenvalue represents the amount of scaling in a direction represented by a corresponding projection matrix.**

The examiner's rejection of claim 22 as set forth in the Office Action is reproduced below for reference:

> **As per claim 22, Thomas and Reyzin demonstrated all the elements as applied to the rejection of dependent claim 19, supra.**
>
> **Thomas and Reyzin disclose a method of distorting an area of an image using affine transformation. It is noted that Thomas and Reyzin do not explicitly disclose the one component of the distortion is a scaling amount, and the instructions to cause the computer to decompose the affine transformation further comprise instructions to calculate a pair of eigenvalues of the linear transform matrix, and wherein each eigenvalue represents the amount of scaling in a direction represented by a corresponding projection matrix, however, this is known in the art as taught by Foley. Foley disclose that, in affine transformation, eigenvector of the transformation can be calculated and it value is a scalar multiple of the vector derived from the transformation (page 1108-1109, A.6).**

The cited portion of Foley describes calculating an eigenvalue of a transformation. However, the Thomas, Reyzin and Foley references do not individually or in combination disclose or suggest extracting a scaling component of a distortion by calculating a pair of eigenvalues of the linear transform matrix. Accordingly, claims 22-24 are patentable over the cited references.

Group IX (claims 12, 27 and 33)

Claims 12, 27 and 33 are rejected under 35 U.S.C. § 103(a) as being unpatentable over Thomas in view of Choi. For the purposes of this appeal only, claims 12, 27 and 33 rise and fall together. Claim 12, which is representative of this group, recites:

> **The method of claim 1 wherein a user selects the area for the applying by the location of a virtual brush.**

The examiner's rejection of claim 12 as set forth in the Office Action is reproduced below for reference:

> **As per claim 12, Thomas demonstrated all the elements as applied to the rejection of independent claim1, supra.**
>
> **Thomas discloses a method of distorting an area of an image using affine transformation. It is noted that Thomas does not explicitly disclose a user selects the area for the applying by the location of a virtual brush, however, this is known in the art as taught by Choi et a., hereinafter Choi. Choi discloses a method of transforming a basic shape element of a character by using a virtual brush (Figure 2 where the brush selects area to be transformed).**

Choi discloses a transformation method for generating a new shape of a prescribed size where the important form of the original shape, such as the thickness of the stroke, is maintained from the original shape. Choi does not disclose or suggest providing a virtual brush that a user can use to select an area to which the extracted component is applied. The only mention of a brush in the Choi reference is in column 4, lines 58 to 62, which describes a medial axis of a character as being a curve drawn by the tip of a brush and the radius information of the medial axis transformation being the size information of how much the brush is pressed down.

The Thomas and Choi references do not individually or in combination disclose or suggest the features of claim 12. Accordingly, claims 12, 27, and 33 are patentable over the cited references.

(8) Claims Appendix

1. A method comprising:

in response to user action on a canvas, selecting at least one area of a first image which relates to an area on a distortion grid;

using a plurality of points local to the at least one area to calculate a distortion;

extracting at least one component of the distortion; and

applying the at least one component to a second area of the first image.

2. The method of claim 1 wherein the at least one component of the distortion is one of displacement, rotation, magnification, skew and directional scaling.

3. The method of claim 2 wherein the extracting comprises calculating an affine transform from the plurality of points.

4. The method of claim 3 wherein the extracting further comprises decomposing the affine transform into a translation and a linear transform matrix.

5. The method of claim 3 wherein the extraction of magnification comprises calculating the determinant of a linear transform matrix.

6. The method of claim 3 wherein the extraction of rotation comprises calculating an angle from the elements of a linear transform matrix.

7. The method of claim 3 wherein the extraction of scaling comprises calculating a pair of eigenvalues of a linear transform matrix, and wherein each eigenvalue represents the amount of scaling in a direction represented by a corresponding projection matrix.

8. The method of claim 7 wherein a rotation is removed from the linear transform matrix prior to calculating the pair of eigenvalues.

9. The method of claim 7 wherein a skew is removed from the linear transform matrix prior to calculating the pair of eigenvalues.

10. The method of claim 1 wherein a user selects the at least one component.

11. The method of claim 10 wherein the user selects the at least one component from a menu displayed on a user interface.

12. The method of claim 1 wherein a user selects the area for the applying by the location of a virtual brush.

13. The method of claim 1 wherein the applying is to an entire image.

14. The method of claim 1 wherein the applying is to a second image.

15. The method of claim 14 wherein the second image is different from the first image.

16. A computer program product, disposed in a computer readable medium, having instructions to cause a computer to:

using a plurality of points surrounding a first area of an image related to an area in a distortion grid, calculate at least one component of a distortion at the first area; and

apply the at least one component of the distortion to a second area of the image.

17. The computer program product of claim 16 wherein the at least one component of the distortion is one of displacement, rotation, magnification, skew and directional scaling.

18. The computer program product of claim 17 further comprising instructions to cause a computer to calculate an affine transform from the plurality of points.

19. The computer program product of claim 18 further comprising instructions to cause the computer to decompose the affine transform into a translation and a linear transform matrix.

20. The computer program product of claim 19 wherein the one component of the distortion is a magnification amount, and the instructions to cause the computer to decompose the affine transformation further comprise instructions to calculate the determinant of the linear transform matrix.

21. The computer program product of claim 19 wherein the one component of the distortion is an angular rotation amount, and the instructions to cause the computer to decompose the affine transformation further comprise instructions to calculate an angle from the elements of the linear transform matrix.

22. The computer program product of claim 19 wherein the one component of the distortion is a scaling amount, and the instructions to cause the computer to decompose the affine transformation further comprise instructions to calculate a pair of eigenvalues of the linear transform matrix, and wherein each eigenvalue represents the amount of scaling in a direction represented by a corresponding projection matrix.

23. The computer program product of claim 22 wherein rotation is removed from the linear transform matrix prior to calculating the pair of eigenvalues.

24. The computer program product of claim 22 wherein skew is removed from the linear transform matrix prior to calculating the pair of eigenvalues.

25. The computer program product of claim 16 wherein a user selects the at least one component.

26. The computer program product of claim 25 wherein the user selects the at least one component from a menu displayed on a user interface.

27. The computer program product of claim 16 wherein the area for the applying is selected by a user, responsive to the movement of a virtual brush.

28. The computer program product of claim 16 wherein the component is applied to an entire image.

29. The computer program product of claim 16 wherein the component is applied to a second image.

30. The computer program product of claim 16 wherein the second image is different from the first image.

31. A computer program product having instructions stored in a computer readable medium, containing instructions to cause a computer to:

display a first image on a canvas, the first image being related to an area on a distortion grid;

responsive to an input device controlled by a user, select an area of the first image;

responsive to a selection by the user from a menu, extract at least one component of a distortion from the area; and

responsive to movement and location of a cursor controlled by the user, apply the at least one component to a second area of the first image.

32. The computer program product of claim 31 wherein the input device is a mouse.

33. The computer program product of claim 32 where the cursor comprises a virtual paintbrush.

34. The computer program product of claim 31 wherein the at least one component of the distortion is one of displacement, rotation, magnification, skew and directional scaling.

35. The computer program product of claim 31 further comprising instructions to cause a computer to calculate an affine transform from the plurality of points.

36. The computer program product of claim 35 further comprising instructions to cause the computer to decompose the affine transform into a translation and a linear transform matrix.

37. The computer program product of claim 31 wherein the components is applied to an entire image.

38. The computer program product of claim 31 wherein the component is applied to a second image.

39. The computer program product of claim 31 wherein the second image is different from the first image.

(9) Evidence Appendix

The evidence on the following ten (10) pages includes a marked up version of the Thomas reference that was cited by the examiner in the Office Action mailed on October 1, 2003.

Animating Direct Manipulation Interfaces

Bruce H. Thomas
Advanced Computing Research Centre
School of Computer and Information Science
University of South Australia
The Levels, SA, Australia 5095
mabht@graceland.Levels.UniSA.Edu.Au

Paul Calder
Department of Computer Science
The Flinders University of South Australia
GPO Box 2100, Adelaide, Australia 5001
calder@cs.flinders.edu.au

ABSTRACT

If judiciously applied, the techniques of cartoon animation can enhance the illusion of direct manipulation that many human computer interfaces strive to present. In particular, animation can convey a feeling of substance in the objects that a user manipulates, strengthening the sense that real work is being done. This paper suggests some techniques that application programmers can use to animate direct manipulation interfaces, and it describes tools that programmers can use to easily incorporate the effects into their code.

Our approach is based on suggesting a range of animation effects by distorting the view of the manipulated object. To explore the idea, we added a warping transformation capability to the InterViews user interface toolkit and used the new transformation to build a simple drawing editor that uses animated feedback. The editor demonstrates the effectiveness of the animation for simple operations, and it shows that the technique is practical even on standard workstation hardware.

KEYWORDS: animation, direct manipulation, graphical interfaces, warp transformation, user interface toolkits, InterViews

INTRODUCTION

Like other recent workers [2], we have set out to explore how techniques borrowed from cartoons and computer animation can enhance the experience of interacting with a computer. However, we are not concerned here with portraying animated data, as would be the case when operating animation editing tools (such as an algorithm animation editor [15]) or when using animation to supplement the presentation of otherwise static information (such as an animated help system [17]). Rather, we wish to apply animation to the interface itself—to enhance or augment the effectiveness of human interaction with applications that present a graphical interface.

UIST 95 Pittsburgh PA USA

The Problem

In his seminal paper [14], Shneiderman identified the essential ingredients of direct manipulation: immediate response to actions, incremental changes, and reversible effects. Properly applied, these characteristics can convince users that they are directly interacting with application data, even though they know it's all an illusion. Central to this illusion, which Brenda Laurel calls suspension of disbelief [9], is the feedback that the application generates to track the manipulation.

Consider the common operation of moving an object by dragging. The application must provide continuous feedback during the drag so that the user knows where the object is at all times. Simple feedback techniques based on drawing object outlines are cheap to implement but tend to give the impression that the operation is happening to a surrogate object rather than the real thing. More ambitious schemes such as drawing the full-featured object during the drag gives the object solidity but can still fail to convey a sense of substance; somehow it seems *too easy* to manipulate the object.

To be convincing, solid-seeming objects must do more than look solid; they must also *feel* solid. One approach widely used by cartoon animators [10] is to use techniques that mimic physical effects such as inertia and friction to reinforce the illusion of substance. Cartoon objects start moving slowly and come to a gradual stop, they move in curves rather than along straight lines, and they stretch and squash to suggest interaction with their environment.

Although we have not conducted formal user trials, our informal experience is that these same techniques can enhance the impression of substance when interacting with application objects. For example, if an object is dragged by a corner, it might distort slightly as if it had inertia; if it collides with other objects, they might wiggle as if bumped. The overall effect is that the user has a stronger sense that "things are happening" and

that real work is being done.

Adding animation to the feedback also allows the application to display more information about the interaction. In addition to the current state of the objects being manipulated, the animation can also suggest the direction and rate of change caused by a user's actions, or it can imply the presence of constraints that prevent the object from changing. The extra information gives the user a better understanding of the results of an action on an object.

These effects can improve the user's satisfaction with the interface, but they add to the application programmer's task. The problem, then, is twofold: to suggest how animation can enhance the interactive experience, and to provide tools that programmers can use to easily incorporate the effects into their interfaces.

Structure of the Paper

This paper describes how we have built support for animating certain kinds of interactions into the InterViews toolkit [11]. First we describe our approach to the problem and illustrate the overall effect of our techniques in the context of a simple drawing editor. Then we discuss the additions and changes we made to the InterViews toolkit components, suggest how the new and modified components can be used to create animated interactions, and briefly outline our implementation to show that the techniques can be successfully applied without special purpose graphics hardware. Finally, we describe other work that has considered animation in user interfaces.

Research Context

This work is part of the Prosodic Interfaces project, an ongoing project that aims to provide a set of tools and techniques that can improve the prosody of human computer interfaces. In addition to animation [19, 20], the Prosodic Interfaces project is concerned with sounds and gestures. Audible reinforcement can improve engagement with the interface and strengthen the sense of direct manipulation by providing contextual cues and a sense of presence. Gestures can provide a smoother and more expressive way of issuing commands and specifying operations. Together, these three techniques will extend the repertoire of tools that interface programmers can use to enrich their creations.

ANIMATING INTERACTION

Laybourne's *Animation Book* [10], outlines a basic set of requirements for making a successful animation. Although developed in the context of film animation, the basic principles also apply to computer animation [8]; in both disciplines, animations are created by displaying sequences of images in rapid succession.

For interface objects that move spontaneously or along predefined paths, the application of Laybourne's techniques is straightforward (Chang and Ungar [5] provide some excellent examples). However in the context of direct manipulation, where users are in control of object movements, application programmers are faced with a dilemma: the need for immediate and accurate response to user actions is at odds with the need for objects to have realistic-seeming behaviour. If a user tries to drag an object rapidly across the screen, for example, how should the application represent the inertia of an object resisting its movement? Or if a user tries to move an object that is fixed in position, how should the application suggest the restraint?

Our Approach

Our approach is to consider how a cartoon animator would depict such behaviour. For example, to suggest movement caused by someone dragging an object, an animator could show the object distorted in the direction of the pull. To suggest an attempt to move a fixed object, the animator could show the object leaning in the direction of the pull while its base stays fixed in place. With techniques like these, an application can create the illusion of a greater sense of substance for its objects while still allowing users to feel that they are in control.

Chang and Ungar list three principles that apply to cartoon animation: the principles of solidity, exaggeration, and reinforcement. To these, we add two more that should also apply when animating a direct manipulation interface:

A

1. The **principle of attachment** states that the object being manipulated should at all times remain attached to the pointer.
2. The **principle of reluctance** states that objects are, in general, reluctant to change.

Adherence to the principle of attachment maintains the impression that the user is always in control of the action. Adherence to the principle of reluctance reinforces the illusion of substance by suggesting that changing objects requires effort on the part of the user.

Demonstrating the Effect

We have applied these principles to the task of animating the interaction with graphical objects in a simple drawing editor. The editor supports the creation of simple figures such as lines and polygons, and it allows simple editing operations such as moving, scaling, and rotating. Figure 1 shows a screen image of the editor being used to move an object.

D

Figure 2 shows how the editor animates a move operation on a rectangle. Of course, these static images cannot fully convey the dynamic feel of the interaction; we

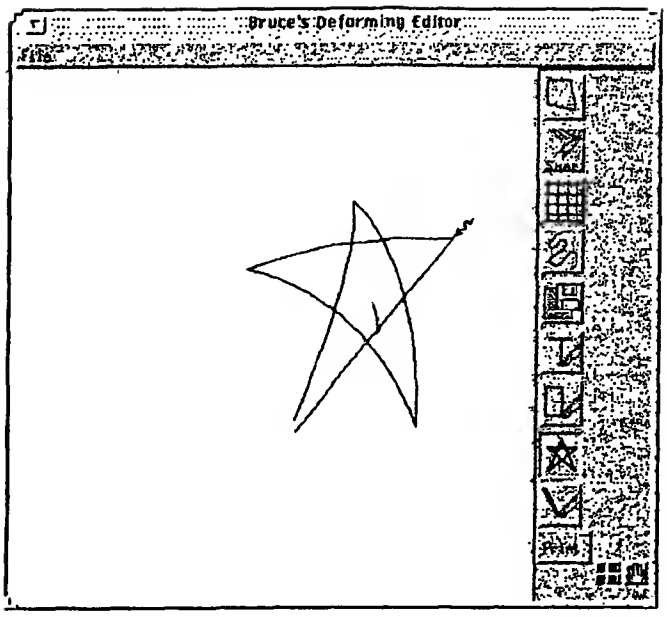


Figure 1: The animated editor in action

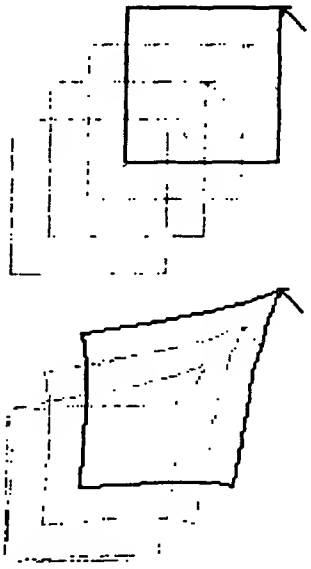


Figure 2: Animating a move operation

have superimposed several successive frames from the interaction to suggest the effect. The top of the figure shows the operation without the animation effect; the rectangle merely moves to follow the mouse. The bottom shows the operation with animation; note how the corner of the rectangle stays attached to the mouse point while the bulk of the object lags slightly behind. In operation, the effect is that of manipulating a heavy "rubbery" object that distorts as it is pushed and pulled.

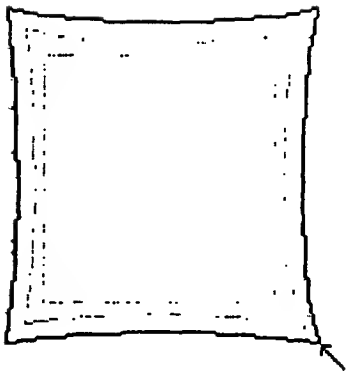


Figure 3: Animating a scaling operation

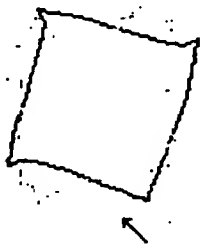


Figure 4: Animating a rotate operation

The same kind of effect works for other common editor operations. For example, figure 3 illustrates a scaling operation, and figure 4 shows a rotation. In each case, the editor uses the same strategy to animate the interaction; the part of the object that is "grabbed" is controlled by the mouse, while the bulk of the object lags behind. (For the scaling operation, we use an effect that suggests that all corners are simultaneously under the control of the mouse.)

The editor also supports simple constraining effects that show how animation can convey extra information about the interaction. Consider, for example, an attempt to move an object that is fixed in place. One response to this attempt might simply be to prevent the object from following the mouse. However, this lack of visible feedback might be misinterpreted as the result of a failure to "grasp" the object correctly. A user might make several attempts at the operation before realising the true cause of the lack of response. Another strategy might be to allow the object to follow the mouse, but then to snap it back to its original place when released. This approach avoids the problem with lack of feedback, but can lead to surprises when a carefully placed object suddenly jumps back.

Figure 5 shows how our editor avoids both problems using animation. As the user attempts to drag the pinned

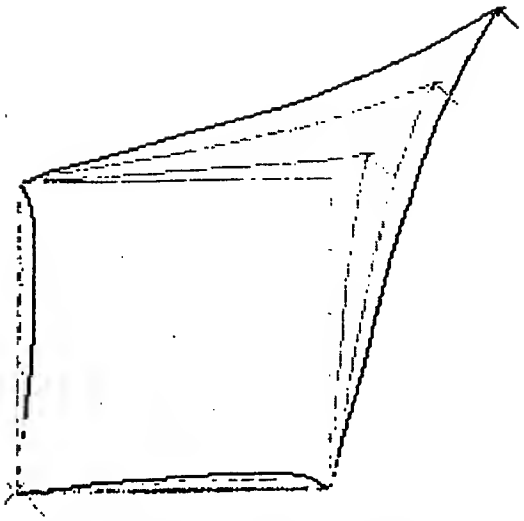


Figure 5: Attempting to move a pinned object

object, the grab point stays attached to the mouse but the bulk of the object stays fixed. The effect is as if user is pulling on a corner of an object that is anchored in place. The feedback makes it clear that the user is attempting to move the object, but that the attempt will not succeed. When the grab point is released, the object will spring back to its original shape.

The same effect can suggest other constraints on objects. For example, figure 6 shows the effect of moving an object under the influence of a "gravity" field that causes the object to snap to a regular grid. As the user pulls the object away from a grid point, the grasped corner stretches while the object stays fixed. When pulled beyond a certain distance, the gravity suddenly "lets go" and the object snaps to the next grid point. Again, the feedback makes it clear what the current state of the interaction is, what the user must do to achieve the desired result, and what will happen if the object is released at any time.

MAKING IT EASY TO USE

To make the animation effects easy to generate, we incorporated them into an existing user interface toolkit. We chose the InterViews toolkit [11] because it has a powerful graphics model that we could extend to include the new effect.

The Model

One way to suggest interaction with physical objects would be to represent the objects with a physical model and the interaction as forces acting on the objects. Then the responses of the objects could be determined by simulating the physical system. However, our experience is that a much simpler model is sufficient to produce the effect we want.

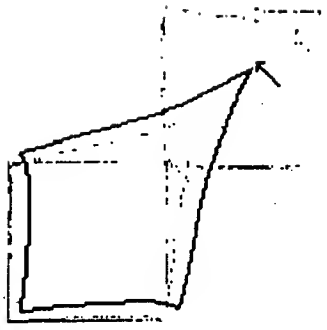


Figure 6: Moving under the influence of "gravity"

Our approach is to draw the manipulated objects as usual, but to warp the coordinate system of the drawing surface so that the view appears distorted. The effect is as if the objects were drawn onto a rubber sheet, and the sheet was then stretched by pulling at certain points. (The technique is widely used to distort images [22].) We implement the warp as a transformation applied to all underlying drawing primitives.

To suggest the effect of moving a rectangle, for example, our editor determines an appropriate warping transformation, applies it to the drawing system, then proceeds to draw the object as a simple rectangle. When the rectangle is displayed on the screen, it will appear distorted.

Apart from its simplicity, this approach has two key advantages when compared with an approach based on a physical model: the code that draws the warped objects is unaffected by (indeed, is unaware of) the presence of the warp; and the effect will apply as well to more complex graphics as to a simple rectangle.

Computing the Warp Vectors

Our warping transformation is characterised by a set of bound vectors that describe the transformations applied to key points in the coordinate space. Transformations for points that do not coincide with vectors are calculated by interpolating between the vectors. Figure 7 illustrates the idea; a later section describes the calculation in detail.

In the example, grid points that coincide with warp vectors are transformed by the specified amounts. Other points are transformed by intermediate amounts that depend on their relative distances from the specified vectors.

To add animation to the interaction, then, the editor need only calculate a set of vectors that characterise an appropriate warp. We have found that, for the kinds

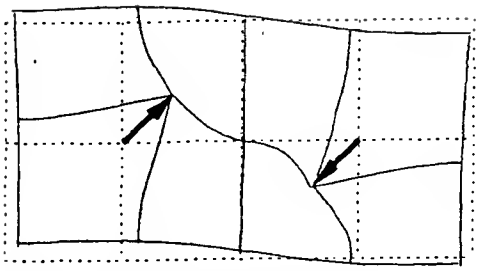


Figure 7: The effect of a set of warp vectors

of shapes and operations we have investigated, a simple algorithm gives very convincing results. For each frame in the interaction, we calculate the warping vectors as follows:

1. Compute the new positions of the vertexes of the shape as usual.
2. Place a zero magnitude vector at the vertex that is grasped by the mouse.
3. Place a vector at each otber vertex that points back to the position of that vertex in the previous frame.

Figure 8 shows the idea for move and rotate operations. The dashed outlines show a rectangle at its old and new positions. The dots represent the zero vectors at the mouse point (in both cases, the rectangle has been grasped by its top right corner), and the arrows show the positions and sizes of the other warp vectors. Finally, the solid strokes show the shape of the distorted rectangle that will be drawn under the warped transformation.

Although the effect does not correspond exactly to a physical model, this simple algorithm gives the impression that the shape is made of elastic material, with weights attached to the vertexes that cause them to lag behind the movement.

Extending InterViews

We implemented our warping model as an extension to the InterViews user interface toolkit. Specifically, we extended the InterViews drawing model by adding a new kind of transformation: a warp transformation. Since all drawing in InterViews is subject to coordinate transformation, this one addition was sufficient to add warping capabilities to all InterViews objects.

The original Transformer object supported only affine transformations such as rotation, scaling, and translation. Figure 9 shows part of the class interface, together with the method that we added to support the new warping transformation.

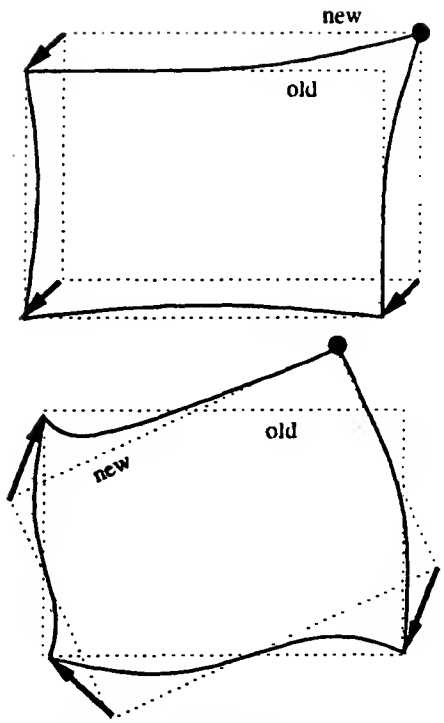


Figure 8: Calculating warp vectors for move and rotate

```
class Transformer ... {
public:
    ...
    void rotate(float a);
    void scale(float x, float y);
    void translate(Coord x, Coord y);

    void warp(Warp& w);       // new method
    ...
    void transform(Coord& x, Coord& y);
    ...
};
```

Figure 9: Extension to Transformer class

```
class Warp {
public:
    void Warp();
    virtual void add_vector (
        Coord x, Coord y,
        Coord vx, Coord vy
    );
    ...
};
```

Figure 10: The new Warp class

```
Transformer t;
Warp w;
w.add_vector(100, 80, 0, 0);
w.add_vector(0, 80, -10, -10);
w.add_vector(0, 0, -10, -10);
w.add_vector(100, 0, -10, -10);
t.warp(&w);
```

Figure 11: Applying a warp for a move operation

The new Warp class represents a set of warp vectors; figure 10 shows the interface. A Warp object initially contains no vectors; calling `Warp::add_vector` adds a vector positioned at the point (x, y) and with components vx and vy. For example, figure 11 shows how the transformation effect of figure 8 could be achieved.

Conceptually, a Transformer represents a succession of discrete mapping steps applied one after the other. For example, a Transformer might represent a scaling followed by a translation followed by a rotation. One characteristic of affine mappings is that they can be computed as matrix products in a homogeneous coordinate system; as a consequence, any sequence of affine mappings can be represented as a single matrix operation. Because of this property, the original Transformer stored only a single 3×2 matrix; successive transformations on the Transformer modified the matrix to reflect the new aggregate operation.

However, the warp mapping is not affine, and so combinations of transformations that involve warping cannot be represented as a single matrix. We modified the implementation of the Transformer class so that it keeps a list of transformation items; successive affine transformations are combined into a single matrix item as before, while warp transformations are represented by individual warp items. The new `Transformer::transform` method iterates through the list of items applying each transformation in turn. This implementation retains the efficiency of the original implementation if the Transformer represents only affine transformations (by far the most common case), but it allows for arbitrary transformations when necessary.

This generalisation of the Transformer class suggests how even more kinds of transformations could be incorporated into the InterViews graphics model, although we have not implemented any at this stage. For example, imagine a "vanishing point" transformation that could be used to suggest perspective drawings, or a "fish-eye" transformation that displays objects under non-uniform scaling.

MAKING IT WORK

In addition to the new Warp class, adding the new warp transformation to InterViews required minor changes to the InterViews drawing code. However, our additions do not affect existing InterViews components, such as glyphs, that use the drawing operations. These components can be used as-is and get the full benefit of the warping operations.

The Warp Transformation Algorithm

Our implementation of the warp transformation computes for each transform point a displacement vector that is a weighted average of the warp vectors[1]. For a point q and a set of warp vectors p_i (each defined by a location p_{il} and a displacement p_{id}), we compute the transformed point q' as follows:

$$q' = q + \sum_{i=1}^{n} w_{qi} p_{id}$$

where

$$w_{qi} = \frac{c_{qi}}{\sum_{j=1}^{n} c_{qj}}$$

and

$$c_{qi} = 1/|q - p_{il}|$$

The coefficient c_{qi} represents the "closeness" of the point q to the transform vector p_{il}; in our current implementation, we compute closeness as the reciprocal of the distance between q and p_{il} (the undefined case when the distance is zero is treated specially). This formulation ensures that the vector weights remain in the range $[0, 1]$, and that the sum of the weights is 1.

Apart from its simplicity and relative efficiency (the algorithm has complexity $O(n)$ for n warp vectors), our implementation of the warp leads to several properties

[1] Our colleague Murk Bottema suggested the algorithm.

that simplify the selection of a suitable set of vectors for a desired warp effect:

1. Points coincident with one of the warp vectors are displaced by the amount of that vector.

2. Points equidistant from two vectors are affected equally by those vectors.

3. In the limit, the combined affect of distant vectors approaches the vector average.

To understand why these properties hold, consider the behaviour of the vector weights as q varies. If q approaches one of the vector locations, p_i, then c_{qi} grows without bound; thus w_{qi} approaches 1 while all other weights approach 0. In the limit, when q is coincident with p_i only that vector will affect the transformation. Conversely, if q becomes very distant from all vectors compared with the distance between the vectors, then all c_{qi} become equal and thus all w_{qi} approach $1/n$.

Drawing

The InterViews drawing model is implemented by the Canvas class, which defines an interface to a set of drawing operations that are similar to those of PostScript [7]. Canvas supports two basic forms of drawing: stroked and filled figures (which are based on PostScript-like paths); and stencils, images, and text (which are based on bitmapped images).

Extending Canvas to draw path-based figures under the warping transformation required only a slight modification. Affine transformations have the property that straight lines remain straight after transformation. To draw a straight path segment, therefore, the original Canvas had merely to transform the segment end points and draw a line between the new points. Because the warping transformation is non-affine, we can no longer make that assumption. Instead, we use an algorithm that recursively bisects the segment until the warped sections can be approximated by straight lines.

Drawing image-based graphics is more difficult. The original Canvas implementation built transformed images and used a standard back-transformation scheme to populate the transformed image with data from the original image. However, we can not use this approach for warp transformations because the warp transformation does not, in general, have an inverse. Instead, we compute an approximation to the warped image by transforming the coordinates of each pixel in the original image and filling the corresponding area in the transformed image. Our current algorithm is slow and produces poor results under some warp conditions, but it is sufficient to demonstrate the effect of the warp. We are investigating solutions to this problem, such as other techniques that have been used for warping images [22].

Performance Issues

Drawing warped graphics is necessarily more expensive than drawing non-warped graphics because of the additional transformation step and because certain assumptions about properties of the distorted view (such as the straightness of "straight" lines) no longer hold. However, our experience is that even a straightforward implementation of the algorithms on standard workstation hardware has adequate performance. For example, when we ran our editor (a standard X11 client) on a Sparcstation 2 with no special graphics hardware, we achieved animation frame rates of around 10 per second for simple objects, even when the X11 connection was carried on a moderately loaded ethernet to a separate server.

Nevertheless, we have investigated a range of approaches for speeding the calculation of the warping transformation and the generation of the drawing algorithms for common conditions. These techniques will be important when drawing complex objects, or to maintain adequate animation frame rates on heavily loaded systems.

One class of warp transformation speedup that we have implemented relies on symmetries in the warp vectors. The idea is to look for equal magnitude warp vectors at the time the transformation is applied. In such cases, some of the computation of the transformation displacement is redundant and can be avoided. Since the warp vectors we use for common manipulations often have equal magnitudes, the benefit can be significant.

For example, consider the warp vectors for a move operation (figure 12). Since $|p_1| = 0$ and $p_2 = p_3 = p_4 = p$, the computation of the transformation simplifies as follows:

$$\begin{aligned} q' &= q + \sum_{i=1}^{n} w_{qi} p_i \\ &= q + w_{q1} p_1 + \sum_{i=2}^{n} w_{qi} p_i \\ &= q + 0 + p \sum_{i=2}^{n} w_{qi} \\ &= q + p(w_{q2} + w_{q3} + w_{w4}) \end{aligned}$$

Although the complexity of the algorithm remains linear in the number of vectors, the simpler computation reduces the number of arithmetic operations. For rotating and scaling, some simplification occurs although the symmetries are fewer and thus the benefit less.

Another technique that we have investigated is the substitution of a more easily drawn shape for the warped

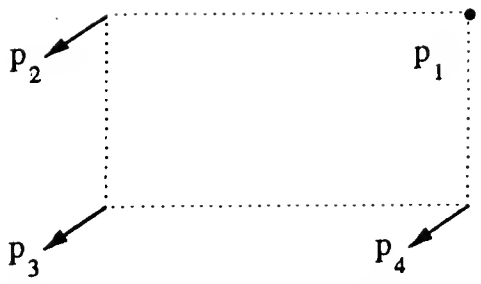


Figure 12: Symmetry of Translation Vectors

figures of path-based graphics [21]. One idea is to substitute a Bezier curve for each segment of a warped path, then draw the curve as if no warping were present. In this way, we only incur the cost of the warping transformation when calculating the control points of the curve, a considerable saving over the usual cost of computing the transformation at many in-between points.

For example, if a warped path segment were represented as a cubic Bezier curve, we would compute only 4 warp transformations (the end points and two control points) instead of the many transformations that the true warping algorithm would typically need.

The approach works reasonably well under "mild" warp conditions, but the approximation of the Bezier curve to the warped curve degenerates for "severe" warps, which leads to unsatisfactory visual artifacts (such as coincident points not remaining coincident). We are investigating modifications to the technique, such as approximating path segments with several Bezier curves, that promise a compromise between performance and accuracy.

RELATED WORK

This section describes other work that has influenced our thinking in relation to the application of animation to interfaces. The material is arranged in three categories: an overview of the basic animation process and how it applies to graphical interfaces; a look at some systems that incorporate animation into their interfaces; and a presentation of other graphical interface toolkits that include support for animation.

Basic Animation Techniques

In her book *Computers as Theatre* [9], Laurel argues that the computer and human are active agents working together to achieve some common goal, and that it is the goal of the designer of an application's interface to facilitate the agents in their effort to collaborate as closely as possible. Laurel advocates the use of dramatic techniques to engage the interest and participation of the human agent. Animation can be used to enhance some of these dramatic effects; for example the effect of *suspended disbelief* is improved if the graphics of an interface move in a smooth and realistic fashion. This smooth motion is closer to the human agent's expectations of the physical world, which leads to an increased willingness to accept what is shown on the screen.

Lasseter considered how to apply basic animation techniques to computer animation in his paper *Principles of Traditional Animation Applied to 3D Computer Animation* [8]. He lists a set of 11 principles: stretch and squash, anticipation, timing, staging, overlapping and follow through action, straight ahead and pose-to-pose action, slow-in and slow-out, arcs, exaggeration, secondary action, and appeal. Along with many others, we have adopted these principles as the basis of our animations.

In their article *Animation at the Interface [2]*, Baecker and Small outline several roles animation can play: animation of structure, animation of process, and animation of function. By their taxonomy, this paper is concerned with animation of function, and with making interfaces more comprehensible. Our technique for animated feedback for direct manipulation is akin to their notion of animation as feedback, although they are chiefly concerned with feedback for system status monitoring whereas we are concerned with feedback for interactive operations.

Systems that use animation

The work of Card, Robertson, and Mackinlay in the area of three dimensional information visualisation relies heavily on good interactive animation. In particular, the cone tree [13] and the perspective wall [12] show how interactive animation can shift the user's processing load from the cognitive to the perceptive system.

The Self programming environment supports cartoon-style animation techniques. Chang and Ungar [5] describe how Self uses solid drawing, squash and stretch, motion blur, dissolves, anticipation, follow through, slow in and slow out, and arcs to strengthen the impression that programmers are manipulating solid objects. One of the goals of the system was its practical implementation on current technology; they use colourmap animation techniques to achieve sufficiently high refresh rates, but only at the expense of a limited choice of colours and types of animation.

Several workers, including Sukaviriya and Foley [17] and Bharat and Sukaviriya [4], have explored animation in the context of an on-line help system. These studies have shown that animation helped convey information more quickly and accurately. Baecker, Small, and Mander used animated icons [3], which offered animated help about the nature and use of the programs attached to the icons.

Finally, HyperCard [1] uses some simple animation techniques to help smooth the transitions between states. The system offers several mechanisms for stack builders to use to animate switching between cards, including dissolves, iris-like resizing, and panning from one card to another.

Toolkits that support animation

Artkit [6] offers application programmers a choice of animation techniques such as simple motion blurring, squash and stretch, arcing trajectories, anticipation and follow through, and slow-in slow-out transitions. In addition, it uses pacing functions to control timing aspects of the animation effects. These functions allow the programmer to map the speed of a graphical object along a curved path to a non-linear time function.

Pacers [18] address the problem of automatically maintaining the frame rate of an animation when there are insufficient computational resources to fully draw complex graphics and respond to user input in a timely fashion. Pacers dynamically adapt the presentation quality of the graphics depending on available resources. For example, if there is insufficient time to draw a solid object, the pacer might omit fine grained detail or draw an object outline instead

The IRIS Inventor [16] is a high-end toolkit with full-scale rendering and real-time animation. However, its primary goal is to support 3D interactive applications on high-performance graphics workstations; it is not designed as a graphical user interface toolkit.

CONCLUSIONS

This work makes three contributions to the field of animation in graphical interfaces:

1. It shows how effects based on cartoon animation principles can be used to enhance the illusion of direct manipulation by strengthening the impression that users are manipulating "real" objects.

2. It defines a new geometric transformation operation based on a set of warp vectors that application programmers can use to incorporate animation effects into their interfaces, and it describes how the new transformation was incorporated into the underlying graphics model of an existing user interface toolkit.

3. It demonstrates that the cost of drawing animated feedback need not be prohibitive, and that good results can be achieved even on standard workstation hardware.

The effects of animation on the user's perception of the interface (like the effects of other aesthetic elements such as colour) can be profound. On the one hand, this influence suggests that great improvements are possible; on the other hand, it warns that equally great disasters can happen. Just as there are good and bad uses of colour, so there are good and bad uses of animation. Inappropriately applied, animation will seem childish and drive users away. But sensibly applied, it can make an interface more graceful and enjoyable to use.

ACKNOWLEDGEMENTS

This work has been supported in part by a grant from the Flinders University Board of Research. We thank the members of the TGI research group at Flinders for providing a stimulating context for this work, and the anonymous reviewers for comments that helped improve the presentation of this paper.

REFERENCES

1. W. Atkinson. *HyperCard.* Apple Computers Inc., 1987.

2. Ronald Baecker and Ian Small. Animation at the interface. In Brenda Laurel, editor, *The Art of Human-Computer Interface Design.* Addison-Wesley Publishing Co., Reading, MA, 1990.

3. Ronald Baecker, Ian Small, and Richard Mander. Bringing icons to life. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, pages 1–6, 1991.

4. Krishna Bharat and Piyawadee Sukaviriya. Animating user interfaces using animation servers. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, pages 69–79, 1991.

5. Bay-Wei Chang and David Unger. Animation: From cartoons to the user interface. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, pages 45–55, 1991.

6. Scott E. Hudson and John T. Stasko. Animation support in a user interface toolkit: Flexible, robust, and reusable abstractions. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, pages 57–67, 1991.

7. Adobe Systems Incorporated. *PostScript Language Reference Manuel.* Addison-Wesley, Reading, Massachusetts, 1985.

8. John Lasseter. Principles of traditional animation applied to 3D computer graphics. In *SIGGRAPH '87*, pages 35–44, Anaheim, CA, July 1987. ACM, ACM Press.

9. Brenda Laurel. *Computers as Theatre.* Addison-Wesley Publishing Co., Reading, MA, 1991.

10. Kit Laybourne. *The Animation Book*. Crown Publishers, Inc., New York, NY, 1979.

11. Mark A. Linton, John M. Vlissides, and Paul R. Calder. Composing user interfaces with InterViews. *IEEE Computer*, pages 8–22, February 1989.

12. Jock D. Mackinlay, George G. Robertson, and Stuart K. Card. The Perspective Wall: Detail and context smoothly integrated. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, pages 173–179, 1991.

13. George G. Robertson, Jock D. Mackinlay, and Stuart K. Card. Cone trees: Animated 3D visualizations of hierarchical information. In *Proceedings of ACM CHI'91 Conference on Human Factors in Computing Systems*, pages 189–194, 1991.

14. Ben Shneiderman. Direct manipulation: A step beyond programming languages. *IEEE Computer*, pages 57–69, August 1983.

15. John T. Stasko. TANGO: A framework and system for algorithm animation. *IEEE Computer*, 23(9):27–39, September 1990.

16. Paul S. Strauss. IRIS Inventor, a 3D graphics toolkit. In Andreas Paepcke, editor, *OOPSLA '93*, pages 192–200, Washington D.C., October 1993. ACM, ACM Press.

17. Piyawadee Sukaviriya and James D. Foley. Coupling a UI framework with automatic generation of context-sensitive animated help. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, pages 152–166, 1990.

18. Steven H. Tang and Mark A. Linton. Pacers: Time-elastic objects. In *Proceedings of the ACM SIGGRAPH Symposium on User Interface Software and Technology*, pages 35–43, 1991.

19. Bruce H. Thomas and Paul R. Calder. An animated widget kit for InterViews. In *Proc. Computer Human Interaction Special Interest Group of the Ergonomics Society of Australia, OZCHI '94*, pages 203–208, Melbourne, Nov 1994.

20. Bruce H. Thomas and Paul R. Calder. Animating widgets in the InterViews toolkit. In *Proc. The 1995 East-West International Conference on Human-Computer Interaction*, pages 88–107, Moscow, Russia, Jul 1995.

21. Bruce H. Thomas and Paul R. Calder. Graphical feedback for direct manipulation of simple polygons. Technical Report CIS-95-007, School of Computer and Information Science, University of South Australia, February 1995.

22. George Wolberg. *Digital Image Warping*. IEEE Computer Society Press, Los Alamitos, CA USA, 1992.

(10) Related Proceedings Appendix

None

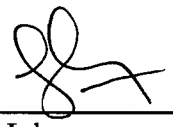
(11) Summary

In addition to any remarks made, all of the dependent claims are patentable for at least the reasons for which the claims on which they depend are patentable. Canceled claims, if any, have been canceled without prejudice or disclaimer.

Any circumstance in which the applicant has (a) addressed certain comments of the Examiner does not mean that the applicant concedes other comments of the Examiner, (b) made arguments for the patentability of some claims does not mean that there are not other good reasons for patentability of those claims and other claims, or (c) amended or canceled a claim does not mean that the applicant concedes any of the Examiner's positions with respect to that claim or other claims.

A check in the amount of $340.00 for the brief fee was submitted on October 7, 2004 with the Appeal Brief. No fee is believed due. Please apply any other charges or credits to Deposit Account No. 06-1050.

Respectfully submitted,



Date: 3/8/06

Mandy Jubang
Reg. No. 45,884

Customer No.: 021876
Telephone: (617) 542-5070
Facsimile: (617) 542-8906
21281665.doc

Attorney's Docket No.	Express Mail Label No.	Mailing Date	For PTO Use Only
07844-495001		October 7, 2004	Do Not Mark in This Area
Application No. 09/996,200	Filing Date November 28, 2001	Attorney/Secretary Init MZJ/cxc	

Title of the Invention
A TOOL FOR EXTRACTING AND MANIPULATING COMPONENTS OF WARPING TRANSFORMS

Applicant
Todor Georgiev

Client Reference No.
P459

Enclosures
·Checks in the amount of $770.00
·Appeal Brief (in triplicate, 21 pages)
·Petition for Extension of Time (2 months)
·Other: Appendix including marked up version of Thomas reference (10 pages)